APPEAL TO THE BOARD OF PATENT APPEALS AND INTERFERENCES

Group Art Unit: 2186
Examiner: Pierre Miche Bataille

January 3, 2007

Customer Assg. No.: 027516
Serial No.: 10/620,406
Filed: 7/16/2003
In re Application of: Vartti et al.
Title: Programmable Cache Management System and Method
Docket No.: RA-5623

APPELLANT'S BRIEF

MS Appeal Brief - Patents
Commissioner of Patents
P.O. Box 1450
Alexandria, VA 22313-1450

This brief is being submitted on January 3, 2007, which is within two (2) months of November 3, 2006, the date of mailing of the Notice of Panel Decision from Pre-Appeal Brief Review. This brief, which is filed in triplicate, is transmitted with the required fees. A petition for a **ONE-MONTH EXTENSION** of time under 37 CFR 1.136(a) is attached herewith. Permission is hereby granted to charge deposit account number 19-3790 for the required fees, and any errors in fee calculation. Appellant requests that this Appeal Brief be made of record and fully considered.

CERTIFICATE OF MAILING (37 CFR 1.8(a))
I hereby certify that this correspondence is being deposited with the United States Postal Service with sufficient postage as first class mail, in an envelope addressed to: Box MS Appeal Brief - Patents, Commissioner of Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date shown below:

Emily Vogt **January 3, 2007**
(Emily Vogt)

01/05/2007 CCHAU1 00000083 193790 10620406
01 FC:1402 500.00 DA

TABLE OF CONTENTS

Real Party In Interest

The real party in interest is Unisys Corporation, with an address as follows:

Unisys Corporation
Township Line and Union Meeting Roads
Blue Bell, Pennsylvania 19424

Unisys Corporation is the real party in interest through an assignment from the inventors of their entire interests, having a recordation date of 7/16/2003 at Reel/Frame 014319/0156.

Related Pending Appeals or Interferences

There are no pending appeals or interferences related to the subject Appeal.

Status of Claims

Claims 1-37 remain pending, and stand finally rejected.

Status of the Amendments

A first amendment was filed on March 1, 2006 to amend Claims 1, 2, 16, 17, 25, 28, 30. This amendment has been entered.

No further amendments have been submitted or entered.

A clean copy of Claims 1-37, as amended, is provided as Appendix A.

Summary of the Invention

Applicants' invention provides a programmable mechanism to control a cache and associated tag logic. In one embodiment, the cache resides within a processing node that includes one or more instruction processors (IPs) for executing instructions. This processing node is coupled to a main memory. Data may be retrieved from main memory and provided to the processing node for use by one or more of the IPs in that node. When this occurs, one or more programmable indicators are used to determine whether the tag logic of the cache will be updated to track the presence of that data within the processing node. One or more other programmable indicators may be used to determine whether the data will be stored within the cache.

The foregoing may be described in more detail in reference to Applicants' drawings. In Figure 1, processing nodes 120A and 120B are coupled to the main memory 100.[1] Each processing node includes one or more IPs to execute instructions, such as IPs 110A – 110D of processing node 120A. Each processing node further includes control logic that is referred to as a processor node director (PND) 102. Each PND includes a cache 206 and associated cache tag logic 204, as shown in Applicants' Figure 2.

When an IP makes a request to read data, and if it is determined that the data is not already resident within the processing node, the request is forwarded to the main memory. In response, the main memory will return the requested data to the PND of the processing node. The PND then determines how that data is to be handled. In one embodiment, the various ways the PND may handle the data include the following:

[1]Applicants' main memory is also referred to as the Storage Coherency Director, or SCD, as set forth in Applicants' Specification ("Specification") p. 9 ll. 16-17.

a.) provide the data to the requesting IP, store the data in the cache 206, and update the cache tags 204[2];

b.) provide the data to the requesting IP, update the cache tags to indicate presence of the data in the processing node, but do not store the data within the cache[3]; or

c.) provide the data to the requesting IP, but do not update the cache tags and do not store the data in the cache[4].

The above discussion focuses on how the PND operates when data is received from main memory. The PND may also receive data from an IP as well as from the main memory. For instance, an IP may update data in its private cache and then provide that data to the PND. In a manner that is similar to that described above, the PND will determine whether to store this data in cache and/or update the cache tags to track the presence of this data within the processing node. In some cases, the data is merely forwarded to main memory without doing either of these things.

The way in which the PND handles data is selected using programmable indicators. These indicators are stored within a programmable storage device shown as cache control store RAM 216 of Applicants' Figure 2. In one particular embodiment, the indicators are stored within a storage mode field 218 of the cache

[2] See, for example, Default Replacement Mode described in pp. 22-26.
[3] See, for example, Selective Replacement described in pp. 27-29.
[4] Store-Through Mode described in pp. 30-31 or Cache-Not-Present Mode described in pp. 31-34.

control store RAM.[5] These indicators select a mode of operation that determines when data is to be stored to cache and/or when cache tags are to be updated to track the presence of the data in the processing node. For example, in a comprehensive replacement mode, generally all data is stored to cache and tracked by the cache tags. In contrast, in a selective replacement mode, the presence of the data within the processor node may be tracked by the tags, but that data is not necessarily stored to the cache. In a cache-not-present mode, data is neither stored to cache nor tracked by the cache tags.

In another embodiment, one or more other programmable fields are used instead of the storage mode field 218 to control which data is stored to cache and/or when that data is tracked by the cache tags. Such fields include a programmable response type field 282, a programmable processor operation type field 284, and a programmable processor field 286. Use of these fields is described in detail in Applicants' Specification.[6]

Applicants' invention allows a user to tailor operation of the system to individual requirements. For instance, a user may desire a less expensive system that does not contain cache logic 106. In this case, the system will be programmed to operate without use of this cache or the associated tags[7]. In a second scenario, a user may be running an application wherein IPs are performing a large number of updates to memory data in their respective private caches. In this case, the system will operate more efficiently if data obtained from main memory is provided directly

[5] Specification p. 21 ll. 20-23.
[6] Specification p. 32 l. 5 – p. 34 l. 17.
[7] Specification p. 34 l. 23 – p. 35 l. 3.

to a requesting IP without being stored in the shared cache 106[8]. During this type of operation, the mode is set so that the cache tags track the presence of the data in the processing node, but the data is not stored to the shared cache 106 upon delivery from main memory. In yet another scenario, it may be desirable to track all data via the tags and store all data in the shared cache.[9]

Some of the aspects of Applicants' invention that are discussed above are described in Applicants' representative Claim 1 as follows:

> A memory system, comprising:
>
> a programmable storage device[10] to store one or more indicators[11];
>
> a cache[12];
>
> cache tag logic[13]; and
>
> a control circuit[14] coupled to the storage device, the cache, and to the cache tag logic, the control circuit to receive data for possible retention in the cache and to determine, based on the state of the one or more indicators, whether to update the cache tag logic to track the data.

[8] Specification p. 35 ll. 16-21.
[9] See "Mode Selection", Specification pp. 34-37.
[10] Cache control store RAM 216 of Applicants' Fig. 2.
[11] In one embodiment, storage mode field 218 of the cache control store RAM 216 is used for this purpose. In other embodiments, fields 282, 284, and/or 286 are used instead.
[12] Cache 206 of Applicants' Fig. 2.
[13] Cache tag logic 204 of Fig. 2.
[14] Cache control logic 202 of Fig. 2.

Issues

I. Whether Claims 1-14, 16-26, and 28-37 are anticipated under 35 USC §102(b) by U.S. Patent Number 5,913,224 to MacDonald (hereinafter, "MacDonald").

This issue may be divided into the following sub-issues A - F:

A.) Whether the MacDonald memory controller teaches Applicants' one or more indicators.

B.) Whether the MacDonald cache management unit teaches Applicants' control circuit and/or indicators.

C.) Whether the embodiment of MacDonald Figure 5 teaches Applicants' control circuit and/or indicators.

D.) Whether MacDonald Figure 4 teaches Applicants' control circuit and/or indicators.

E.) Whether the MacDonald lock bits teach Applicants' programmable storage device.

F.) Whether the MacDonald translation look-aside buffer teaches Applicants' programmable storage device.

II. Whether Claims 15 and 27 are unpatentable over MacDonald under 35 USC §103(a).

Grouping of the Claims

Claims 1-14, 16-26, and 28-37 stand or fall together; and

Claims 15 and 27 stand or fall together.

Arguments

I. Whether Claims 1-14, 16-26, and 28-37 are anticipated under 35 USC §102(b) by U.S. Patent Number 5,913,224 to MacDonald (hereinafter, "MacDonald").

Before considering this issue in detail, the MacDonald system is summarized for discussion purposes.

MacDonald discloses a cache configuration which normally operates using a least-recently-used (LRU) replacement mechanism. That is, during normal system operation, when data is received from system memory, a cache management unit stores this new data into cache by aging out other data that is determined to be least-recently used. This is described in MacDonald as follows:

> "...under normal cache operation...the least most recently used cached data is replaced by presently accessed data."[15]

According to the MacDonald system, this normal LRU operation can be over-ridden by a processor when that processor is reading "real-time" code. This occurs as follows:

> "An output signal from the processor during the reading of the real-time code indicates to the cache subsystem the real-time nature of the code. In response, the cache subsystem locks the code into the cache preventing overwriting the code with more recently used data. The real-time code is locked into cache by setting a lock bit associated with each line of cache containing the real-time code."[16]

This locking of the code in the cache prevents the real-time code from being aged out of cache according to the normal LRU replacement mechanism. This, in turn, allows the processor to complete code execution without accessing system memory, as follows:

> "Once locked, the line of data cannot be overwritten pursuant to normal cache behavior in which the least recently used cache line is overwritten by new data..."[17]

The embodiment described above uses lock bits to lock real-time code into cache. In an alternative embodiment, the MacDonald system uses address registers rather than lock bits to implement the locking mechanism. This address registers define which memory addresses contain real-time code. If locking

[15] MacDonald Abstract ll. 11-13. See also MacDonald col. 5 ll. 60-63.
[16] MacDonald col. 3 ll. 5-12.
[17] MacDonald col. 6 ll. 6-9.

capability is enabled, and if the processor makes a read request to an address of memory that stores real-time code as determined by the contents of an address register, the retrieved real-time code is stored in a dedicated cache data way that is not subject to an LRU replacement mechanism. Otherwise, the retrieved data/code is stored to a different cache data way that is subject to an LRU replacement mechanism.[18]

In MacDonald, regardless of which embodiment is being considered, or whether real-time code or some other type of code/data is received from main memory, *all code/data provided to the MacDonald cache is stored to the cache and tracked by the MacDonald cache tags*. In other words, MacDonald does not teach, or even suggest, any scenario wherein data that is provided to the cache will not be tracked by the cache tags, or will not be stored to the cache. It follows that MacDonald does not teach Applicants' indicators that are used to determine whether the cache tags will track data that is provided to the cache. This is described in detail below.

With the foregoing summary available for discussion purposes, a more detailed analysis of MacDonald is provided in reference to Applicants' representative Claim 1. As discussed above, Applicants' Claim 1 includes a control circuit that receives data that may possibly be retained in Applicants' cache. The control circuit uses the state of one or more indicators stored within a programmable storage device to determine whether to update the cache tag logic to track the data.[19]

[18] MacDonald col. 3 ll. 24-40.
[19] Applicants' Claim 1 ll. 2 and 5-8.

In regard to Applicants' control circuit and indicators, the Examiner, quoting MacDonald, asserts:

> "Column 5 lines 3-8 – State that the cache management unit directs data transfers in and out of the L2 cache, and orchestrates the transfer of data, address and control signals between local bus and system memory, and also includes a memory controller (a.k.a. indicator))"[20]

As best understood, it appears the Examiner is hereby asserting:

A.) the MacDonald memory controller teaches Applicants' one or more indicators; and

B.) the MacDonald cache management unit 202 teaches Applicants' control circuit.

These and other considerations are discussed in turn in regard to the following sub-issues.

A.) Whether the MacDonald memory controller teaches Applicants' one or more indicators.

MacDonald does not describe the memory controller in any detail, and this element does not appear in any of the MacDonald drawings. In fact, the only reference in the entire MacDonald description to the memory controller is as follows:

> "Cache management unit 202 preferably includes a memory controller for providing access to L2 cache memory 201. The memory controller may be any one of a number of commonly known memory controllers compatible with the selected CPU core 110 and overall computer architecture. Such a memory controller may be located as part of the processor 101."[21]

[20] Final Rejection of the Claims dated 4/25/2006 (hereinafter "Final Rejection"), p. 3 ll. 1-5.

Beyond this brief reference, MacDonald is silent about how the memory controller operates. As may be appreciated, this reference does not in any way teach, or even allude to, any indicators. This passage likewise does not teach any programmable storage device to store such indicators, as described by Applicants' Claim 1. Thus, the Examiner's apparent assertion that the memory controller teaches Applicants' indicators is not understood. This memory controller does not teach, or even suggest, Applicants' indicators as described in Applicants' Claim 1.

<u>B.) Whether the MacDonald cache management unit teaches Applicants' control circuit and/or indicators.</u>

The cache management unit 202 is included in the first MacDonald embodiment, which is illustrated in MacDonald Figure 2. As discussed above, that embodiment employs lock bits to lock real-time code into the L2 cache. In the description of Figure 2, <u>any and all data</u> received from system memory by the cache management unit (including real-time code and any other code/data) is <u>stored in cache</u>. For instance, when real-time code is retrieved from memory, MacDonald states that the processor stores the code in cache and locks the cache line so that code is ineligible for replacement according to the LRU algorithm.[22] Likewise, when any other code/data is retrieved from memory, "normal cache behavior" is employed wherein that code/data is likewise stored to cache. However, in this case, the lock bits are not activated such that this code/data will be eligible for replacement

[21] MacDonald col. 5 ll. 6-13.
[22] MacDonald col. 6 ll. 29-36, emphasis added.

according to the normal LRU algorithm.[23] Because all information that is provided to the cache is stored within the cache, it follows that the cache tags must always be updated to reflect the presence of the data within the cache.

Further in support of the foregoing, it may be noted that the very use of the lock bits indicates that both real-time code and nonreal-time code/data are cached. That is, if only real-time code were cached by the MacDonald system, all locations in cache that stored valid information, as determined by valid bits[24], could be considered locked. In this case, the use of the lock bits would be entirely unnecessary.

MacDonald Figure 2 further supports the proposition that all code/data is stored to cache. In particular, Figure 2 illustrates the system memory 300 as being coupled directly to the L2 cache subsystem via bus interface 235. According to that configuration, it appears any data received from the system memory must be stored to cache before being provided to processor 101. There is no interface whereby code/data is provided directly from system memory to the processor.

To summarize, nothing in MacDonald describes the cache management unit 202 as caching, and/or as tracking via the tags, a selected sub-set of received code/data. In particular, nothing in MacDonald describes the cache management unit 202 as using indicators that are retained in any programmable device to determine whether to update cache tags to track received data. Instead, MacDonald describes the cache management unit 202 as caching all received data.

[23] MacDonald col. 6 ll. 7-10, emphasis added.
[24] MacDonald col. 6 ll.34-40.

Therefore, the MacDonald cache management unit 202 does not teach Applicants' control circuit and/or indicators as described in Applicants' Claim 1.

C.) Whether the MacDonald embodiment shown in MacDonald Figure 5 teaches Applicants' control circuit and/or indicators.

The foregoing discussion focuses on the Examiner's assertion that the MacDonald cache management unit 202 teaches Applicants' control circuit. This cache management unit 202 is included in a first embodiment of the MacDonald system shown in Figure 2. As previously discussed, this embodiment utilizes lock bits to lock real-time code into cache. For completeness, and although the Examiner does not raise this issue, the additional MacDonald embodiment shown in Figure 5 is also considered in reference to Applicants' Claim 1.

The second MacDonald embodiment provides a dedicated cache way 1 to store all real-time code, and a different cache way 0 to store all other code/data. Thus, when real-time code is retrieved from system memory, that code is

> "...stored in data way 1 without replacing any other real-time code."[25]

When other (nonreal-time) code/data is retrieved from system memory, that code/data is stored

> "...in data way 0 preferably according to the least recently used algorithm described previously."[26]

[25] MacDonald col. 10 ll. 34-36, emphasis added.
[26] MacDonald col. 10 ll. 53-55, emphasis added.

As may be appreciated from the foregoing, the second MacDonald embodiment operates in a manner similar to the first embodiment. That is, all code/data retrieved from system memory is stored within the cache regardless of whether that code/data is real-time code, or some other code/data. Therefore, all such data must likewise be tracked by the cache tags. There is no teaching regarding the MacDonald second embodiment wherein a programmable storage device stores indicators that determine whether data received by the cache will be tracked by cache tags. Thus, the second MacDonald embodiment does not teach Applicants' control circuit and/or indicators of Claim 1.

D.) Whether the flow diagram of MacDonald Figure 4 teaches Applicants' control circuit and/or one or more indicators.

Although the Examiner does not raise this issue, for completeness sake it is next considered whether anything in the flow diagram of Figure 4 teaches Applicants' control circuit and/or indicators.

MacDonald Figure 4 is a flow diagram that summarizes the operation of the first MacDonald embodiment. Recall that according to this embodiment, real-time code is locked in cache by activating lock bits. Nonreal-time code/data is cached, but not locked, and is therefore eligible for replacement using an LRU algorithm. In accordance with this embodiment, MacDonald Figure 4 includes a decision step 420 for determining whether retrieved code is real-time code. If so, the code is locked into the cache, as shown in step 440. If the code is not real-time, MacDonald describes that "known protocols" are employed in step 425 to handle this code.[27] These "known protocols", also described elsewhere in MacDonald as "normal cache behavior", involve storing, but not locking, the code so that it is eligible for replacement using an LRU algorithm.[28] In either case, MacDonald describes the code as being replaced in cache, requiring that the tags likewise be updated to track that code.

As was discussed above, the very use of the locking mechanism in step 440 of Figure 4 indicates that <u>all</u> code/data is cached. In other words, if only real-time code were ever cached, every cache location that stored valid information as indicated by an associated valid bit[29] could be considered real-time code. All cached information could therefore be handled as though it were locked, and the use of lock bits would be entirely superfluous. This is not the case, however. That is, in MacDonald, the lock bits are needed to identify the real-time code because caching occurs for all code/data that is provided to cache.

To summarize, nothing in MacDonald Figure 4 teaches Applicants' control circuit or indicators to determine whether cache tags are to be updated to track data received by the cache.

<u>E.) Whether the MacDonald lock bits teach Applicants' programmable storage device.</u>

[27] MacDonald col. 8 l. 20.
[28] MacDonald col. 8 l. 20 and col. 6 ll. 7-10.
[29] MacDonald col. 5 ll. 34-36.

Next, the Examiner's assertions regarding Applicants' programmable storage device are considered. The Examiner states that the MacDonald programmable cache teaches Applicants' programmable storage device.[30] This assertion was made in reference to the first MacDonald embodiment that utilizes programmable lock bits[31] to override the normal LRU replacement mechanism of the cache. Thus, according to one interpretation of this rejection, the Examiner may be asserting that the MacDonald programmable lock bits 230[32] teach Applicants' programmable storage device and indicators.

The MacDonald lock bits do not teach Applicants' indicators for at least the reason that the lock bits do not control whether the MacDonald cache tags are updated. As discussed above, the MacDonald <u>cache tags are updated for any and all code/data received from system memory</u> regardless of whether that code/data is real-time code. Therefore, the MacDonald lock bits do not teach Applicants' programmable storage device or Applicants' stored indicators that determine whether the cache tags are updated.

<u>F.) Whether the MacDonald translation look-aside buffer teaches Applicants' programmable storage device.</u>

Finally, when the Examiner cites the MacDonald programmable cache as teaching Applicants' programmable storage device, it is possible that the Examiner is more specifically alluding to the MacDonald translation look-aside buffer (TLB) as

[30] Final Rejection p. 2, section 4, ll. 5-6.
[31] Final Rejection p. 2, which rejects Claim 1 based on the MacDonald first embodiment.

teaching this programmable storage device. The TLB contains real-time code bits 153 that specify whether information stored at an associated physical page address in system memory is real-time code.[33] The processor uses the real-time code bits to determine whether to assert its lock signal and indicate that retrieved information is to be locked in the cache.[34]

The TLB and the real-time code bits do not teach Applicants' programmable storage device and indicators for at least the reason that the MacDonald real-time code bits are not used to determine whether to update cache tags. As discussed above, the MacDonald cache tags are updated for <u>all</u> received code/data, regardless of whether the code/data is real-time code, and regardless of the state of the real-time code bits.

In conclusion, MacDonald does not teach each and every element of Applicants' Claim 1 as required by USC §102(b). Specifically, MacDonald does not teach Applicants' control circuit that determines, based on the state of one or more indicators, whether to update cache tags to track received data. Moreover, MacDonald does not teach Applicants' programmable storage device to store the one or more indicators. For at least these reasons, this rejection is improper, and shown be overturned.

The remaining independent Claims 16 and 28 recite similar aspects as those set forth in representative Claim 1. In a manner similar to that described above, MacDonald does not teach each and every element of these additional independent Claims. Likewise, MacDonald does not teach each and every element of the

---

[32] MacDonald Fig. 2.
[33] MacDonald col.4 ll. 50-54 in reference to Figure 2.

various dependent Claims 2-14, 17-26, and 29-37 that depend from a respective one of independent Claims 1, 16 and 28. Therefore, the Examiner has failed to set forth a prima facie case of anticipation for Applicants' pending Claims 1-14, 16-26, and 28-37 as required by 35 USC §102(b), and it is requested that this rejection be overturned.

II. Whether Claims 15 and 27 are unpatentable over MacDonald under 35 USC §103(a.)

Claims 15 and 27 relate to the aspect of Applicants' invention wherein mode switch logic 290 of Figure 2 may automatically re-program one or more of the indicators stored within cache control store RAM 216. This thereby changes the mode in which Applicants' system is executing.

As an example of the use of the mode switch logic 290, assume Applicants' system is operating in selective replacement mode. According to this mode, not all data that is received by the cache is replaced in the cache.[35] For example, if data is provided to the cache from main memory with write privileges, the tags will be updated but the data will not be stored to cache. While operating in this mode, mode switch logic 290 monitors the number of cache misses per unit time. If this number exceeds a programmed threshold value, mode switch logic automatically

[34] MacDonald col. 6 lines46-51.
[35] Specification p. 27 l. 8 – p. 29 l. 18.

modifies the mode of operation to one that causes all data to be both cached and tracked by the tags. This allows the system to operate more efficiently.[36]

Next, the specific language of representative Claim 15 is considered. This Claim is as follows:

> 15. The method of Claim 1, and further including mode switch logic[37] coupled to the programmable storage device[38] to automatically re-program at least one of the indicators in response to monitored conditions occurring within the memory system[39].

The Examiner cites the MacDonald write-back circuit of column 5 lines 47-52 as teaching Applicants' mode switch logic. This passage is as follows:

> "Cache management unit 202 includes an address tag and state logic circuit (not specifically shown) that contains and manages the address tag and state information. A comparator circuit for determining whether a cache hit has occurred, and a snoop write-back circuit that controls the write back of dirty data within L2 cache memory 201 (sic)."[40]

In regard to this passage, the Examiner states:

> "...there is a write-back circuit that replaces data within the L2 cache based on a certain state...Therefore, stating that [Applicants' indicators are re-programmed] automatically does not change the purpose or the functionality of the claimed invention. Therefore, it would have been obvious to one of ordinary skill in the art to enable MacDonald's 'Programmable Cache' to automate the re-programming in order to make the whole programming process faster and more user-friendly."[41]

[36] For example, Specification p. 34 l. 20 – p. 39 l. 14 provides a discussion of mode switch logic.
[37] Mode switch logic 290 of Figure 2.
[38] Cache control store RAM 216 of Figure 2.
[39] Operation of the mode switch logic is described in detail in Applicants' Figure 5.
[40] MacDonald col. 5 ll. 47-52.
[41] Final Rejection, paragraph bridging pp. 11 and 12.

The write-back circuit is mentioned only once in the entire MacDonald disclosure in the passage cited above. Because of the lack of description in MacDonald regarding the write-back circuit, as well as the lack of specificity in the Examiner's assertions, this rejection is difficult to address. However, in attempt to address this rejection, discussion returns to the subject matter of Claim 15. As previously described, this Claim describes mode switch logic that automatically re-programs Applicants' indicators in response to monitored conditions. As discussed at length above, these indicators control whether Applicants' control circuit will update cache tag logic to track received data[42]. The MacDonald system does not teach, or even suggest, any such indicators. Rather, in MacDonald, cache tags are always updated when data is provided to the cache from system memory.

Because MacDonald does not teach, or in any way suggest, the type of indicators described by Applicants' Claim 1, MacDonald does not in any way suggest any circuit for re-programming such indicators. In particular, the MacDonald write-back circuit cited by the Examiner is likely used to store updated data from cache to system memory. Nothing regarding this functionality in any way relates to the re-programming of indicators such as described in Applicants' Claims 1 and 15.

To summarize, nothing in MacDonald in any way suggests mode switch logic to automatically re-program indicators of a type described by Applicants' Claim 1. Therefore, the rejection of representative Claim 15 is improper for this reason, as well as all of the reasons discussed above in regards to Issue I. The rejection of Claim 27, which includes aspects similar to those set forth in Claim 15, is likewise

[42] Claim 1 ll. 5-8.

improper. For these reasons, it is respectfully submitted that Claims 15 and 27 are not unpatentable over MacDonald.

Conclusion and Request for Relief

Applicants' Claims 1-37 are patentable over MacDonald. This reference does not teach each and every element of Applicants' independent Claims 1, 16, and 28 and respective dependent Claims 2-14, 17-26, and 29-37. The Examiner has therefore failed to set forth a prima facie case of anticipation under 35 USC §102(b) in regards to these Claims. Furthermore, MacDonald does not even begin to suggest the additional aspects of dependent Claims 15 and 27, and these Claims are therefore not unpatentable over MacDonald. It is respectfully requested that the rejection of all Claims be overturned, and the Claims be passed to issue.

Respectfully submitted,

Beth L. McMahon 1/3/2007

Beth L. McMahon
Attorney for Applicants
Reg. No. 41,987
Telephone No. 651-635-7893
UNISYS Corporation
M.S. 4773
PO Box 64942
St. Paul, MN 55164-0942

Appendix A

Presented is a clean set of Claims 1-37 as last amended March 1, 2006.

We Claim:

1. (Previously Amended) A memory system, comprising:

a programmable storage device to store one or more indicators;

a cache;

cache tag logic; and

a control circuit coupled to the storage device, the cache, and to the cache tag logic, the control circuit to receive data for possible retention in the cache and to determine, based on the state of the one or more indicators, whether to update the cache tag logic to track the data.

2. (Previously Amended) The memory system of Claim 1, and wherein the control circuit further includes circuits to determine, based on the one or more indicators, whether to store the data to the cache.

3. (Original) The memory system of Claim 2, wherein one of the indicators indicates the cache is not available for use.

4. (Original) The memory system of Claim 2, and further including:

at least one requester coupled to the control circuit to request data from, and store data to, the cache;

a main memory to provide to the cache requested data that is not stored within the cache; and

wherein the control circuit includes a circuit that may replace the data in the cache based on the state of the indicators.

5. (Original) The memory system of Claim 4, wherein the main memory provides data to the cache in response to a request that is any one of multiple request types, wherein at least one of the indicators identifies one or more of the request types, and wherein the control circuit prevents the replacement of the data in the cache if the data was provided in response to any of the identified request types.

6. (Original) The memory system of Claim 4, wherein the one or more request types includes a request type indicating the data will be modified by a requester.

7. (Original) The memory system of Claim 4, wherein at least one of the indicators identifies one or more of the at least one requester, and wherein the control circuit replaces the data in the cache if the data was returned from the main memory in response to a request issued by any of the identified requesters.

8. (Original) The memory system of Claim 4, wherein the main memory provides data to the cache with a response that is any one of multiple response types, wherein at least one of the indicators identifies one or more of the response

types, and wherein the control circuit replaces the data in the cache if the data is returned from the main memory with any of the identified response types

9. (Original) The memory system of Claim 2, and further including at least one requester coupled to the control circuit to return data to the cache tag logic, and wherein the control circuit determines whether to store the returned data to the cache based on the state of at least one of the indicators.

10. (Original) The memory system of Claim 9, wherein the at least one requester returns data to the cache tag logic during an operation that is any one of multiple operation types, wherein the indicators include an indicator to identify one or more of the operation types, and wherein the control circuit stores the returned data to the cache if the returned data is returned during any of the identified operation types

11. (Original) The memory system of Claim 10, wherein the control circuit is further adapted to store the returned data to the cache based, at least in part, on whether a cache hit occurred.

12. (Original) The memory system of Claim 9, and further including a main memory coupled to the control circuit, and wherein the control circuit is adapted to forward the returned data to the main memory based, at least in part, on the state of at least one of the indicators.

13. (Original) The memory system of Claim 12, wherein memory coherency actions may be incomplete for the returned data or for associated data retained by the at least one requester or the cache, and further including a request tracking circuit coupled to the control circuit to prevent the returned data from being forwarded to the main memory until all of the memory coherency actions have been completed for the returned data or for the associated data.

14. (Original) The memory system of Claim 1, wherein the programmable storage device includes circuits to store microcode, and wherein the control circuit is controlled by the microcode..

15. (Original) The method of Claim 1, and further including mode switch logic coupled to the programmable storage device to automatically re-program at least one of the indicators in response to monitored conditions occurring within the memory system.

16. (Previously Amended) A method of controlling a memory system having cache tags to record which data is stored within one or more associated caches, and further having one or more programmable control indicators, comprising:

a.) obtaining data; and

b.) determining whether to update the cache tags to record the data based on the state of one or more of the control indicators.

17. (Previously Amended) The method of Claim 16, and further including determining whether to store the data in a predetermined one of the associated caches based on the state of one or more of the control indicators.

18. (Original) The method of Claim 17, wherein the memory system includes a main memory coupled to the cache tags, and wherein the obtaining step includes:

providing a request for the data to the main memory; and

receiving the data from the main memory.

19. (Original) The method of Claim 18, wherein the request is any one of multiple types, wherein one of the control indicators identifies one or more of the multiple request types, and wherein at least one of the determining steps is performed based, at least in part, upon whether the request is any of the identified response types.

20. (Original) The method of Claim 18, wherein the data is provided from the main memory with a response type that is any one of multiple response types, wherein one of the control indicators identifies one or more of the multiple response types, and wherein at least one of the determining steps is performed based, at least in part, upon whether the request is any of the identified response types.

21. (Original) The method of Claim 18, wherein the memory system is coupled to at least one requester, wherein one of the control indicators identifies one or more of

the at least one requester, and wherein at least one of the determining steps is performed based, at least in part, upon whether the request was initiated by any of the identified requesters.

22. (Original) The method of Claim 17, wherein the memory system is coupled to at least one requester, and wherein step a.) includes obtaining the data from any one of the at least one requester.

23. (Original) The method of Claim 22, wherein the data is obtained during an operation that is any of multiple operation types, wherein one of the control indicators identifies one or more of the operation types, and wherein at least one of the determining steps is based, at least in part, on whether the data is obtained during any of the identified operation types.

24. (Original) The method of Claim 23, wherein at least one of the determining steps is based, at least in part, on whether a cache hit occurs.

25. (Previously Amended) The method of Claim 22, wherein the memory system includes a main memory, and further including providing the data to the main memory instead of storing the data into the predetermined one of the associated caches.

26. (Original) The method of Claim 25, wherein the data is associated with incomplete memory coherency actions, and further including preventing the data from being provided to the main memory until all incomplete memory coherency actions have been completed.

27. (Original) The method of Claim 16, and further comprising:

c.) monitoring conditions within the memory system; and

d.) automatically re-programming at least one of the control indicators based on one or more of the monitored conditions.

28. (Previously Amended) A memory system, comprising:

main memory means for storing data;

cache means for storing a subset of the data; and

programmable storage means for storing control indicators to determine how the subset of the data is to be selected.

29. (Original) The memory system of Claim 28, wherein requests are issued to the main memory to retrieve data from the main memory, and wherein the programmable storage means includes means for selecting the subset of the data based, at least in part, on a type of request that was issued to retrieve the subset of the data from the main memory.

30. (Previously Amended) The memory system of Claim 28, and further including one or more requester means for causing data to be retrieved from the main memory, and wherein the programmable storage means includes means for selecting the subset of the data based, at least in part, on the identity of one or more of the requester means that caused data to be retrieved from the main memory.

31. (Original) The memory system of Claim 28, wherein the main memory means includes means for returning a response type to the cache means with data, and wherein the programmable storage means includes means for selecting the subset of the data based, at least in part, on the response type.

32. (Original) The memory system of Claim 28, and further including requester means for returning data to the cache means, and wherein the programmable storage means includes means for selecting whether data returned by the requester means will be stored to the cache means.

33. (Original) The memory system of Claim 32, wherein the requester means includes means for returning data during any of multiple types of operations, and wherein the programmable storage means includes means for selecting whether returned data will be stored to the cache means based, at least in part, on the type of operation that resulted in return of the data.

34. (Original) The memory system of Claim 32, wherein the programmable storage means includes means for selecting whether data returned by the requester means will be stored to the cache means based, at least in part, on whether a cache miss occurred to the cache means.

35. (Original) The memory system of Claim 28, and further including mode switch means for modifying the state of one or more of the control indicators based on monitored conditions occurring within the memory system.

36. (Original) The memory system of Claim 28, and wherein the cache means includes cache tag means for tracking data that may be stored to the cache means, and wherein the programmable storage means includes means for determining whether to update the cache tag means to track data.

37. (Original) The memory system of Claim 36, wherein the programmable storage means includes means for enabling the tracking by the cache tag means of predetermined data that is not included in the subset of the data stored within the cache means.